Petri Net Representation for the Process Specification Language - Part 1: Manufacture Process Planning

D. Kiritsis, P. Xirouchakis and C. Gunther

CAD/CAM Laboratory (LICP), Department of Mechanical Engineering, Swiss Federal Institute of Technology at Lausanne (EPFL), CH-1015 Lausanne, Switzerland

Abstract

In this paper we propose Petri nets as a representation technique for modelling and analysing process planning activities as they are specified in the Process Specification Language (PSL) project of NIST. The goal of this project is to identify or create a process specification language (PSL) that can be common to all manufacturing applications, generic enough to be decoupled from any given application, and robust enough to be able to represent the necessary process information for any given application. The model discussed in this paper allows the modeling of tasks, their precedence constraints and the required resources (machines, setup and tools) and the corresponding costs within a compact Petri net, named Compact Process Planning net (CPP-net). This model can be easily extended and incorporate time. This part will be discussed in a next paper.

1. INTRODUCTION

Process planning and production planning for manufacturing processes are still most times two distinct sequential off-line activities. *Process plans* are ordered sequences of tasks able to transform raw material to a final part or product. Tasks are chosen taking into account available or potential production means. *Production planners* receive process plans as their input and their task is to schedule the tasks on the machines while respecting the precedence relations given in the process plans.

However, decisions made at the process planning stage, e.g., selection of machines, selection of task sequence, constrain the available choices for optimization on the subsequent production planning phase. On the other hand, process planning and scheduling may have conflicting objectives, such as required technology versus resource usage [Chryssolouris and Chan 1985].

Petri nets have been extensively used for modeling Discrete Event systems and FMS. For a review of Petri net applications in process planning and a complete list of related references see the papers of (Cecil et.al., 1992], [Srihari and Emerson, 1990], [Kiritsis and Porchet, 1995] and [Kiritsis and Xirouchakis 1996]. Various aspects related to the use of Petri nets for the Modeling of manufacturing systems can be found in [Ham and Lu, 1988], [Kruth and Detand,1992], and [Tönshoff et.al. 1991 and 1993].

In the present paper our attention is concentrated into the process planning problem and a new structure of a Petri net model for dynamic process planning is proposed. This model is generic in the sense that its construction is based on a set of standard generic rules and its graphic representation is similar for any part to be processed. With the proposed method two tools are used for model analysis and solutions finding: 1) the simulation tool can show

visually, on the net graph, non-desirable conflict situations. Simulation on Petri nets is performed by firing it from its initial tokening and observing tokens traveling through the net. 2) The reachability analysis tool gives all possible solutions (process plans) dynamically included in the Petri net.

2. THE PROCESS SPECIFICATION LANGUAGE (PSL) PROJECT

In this section, a brief description of the PSL project is given, focusing the process planning aspects. Definitions given here have been taken from the official homepage of the PSL project (http://www.nist.gov/psl/) where the reader may refer for more details, references and official documents. See also [Schlenoff et.al., 1997].

The goal of the NIST Unified Process Specification Language project is to identify or create a process specification language (PSL) that can be common to all manufacturing applications, generic enough to be decoupled from any given application, and robust enough to be able to represent the necessary process information for any given application. Additionally, the PSL should be sufficiently well-defined to ensure complete and correct exchange of process information among established applications. This PSL would facilitate communication between the various applications because they would all "speak the same language", either as their "native" language or a "second" language, for exchange.

The PSL Project reached a pivotal point in Spring, 1997 in having to determine the representation structure for the PSL. Through extensive research and with feedback from colleagues in industry and academia, a set of requirements for specifying process was defined. The project then completed an analysis of existing process representations with respect to these requirements in order to identify constructs and characteristics of these representations which address the requirements. Based on this analysis, the next step was to identify the fundamental foundation for a Unified Process Specification Language, likely based on multiple characteristics and constructs of existing representations.

In this paper we present the CPP-nets as a representation technique of PSL concepts for process planning.

3. PSL REQUIREMENTS FOR PROCESS PLANNING

Process planning

A process plan specifies what raw material or components are needed to produce a product, and what processes and tasks are necessary to transform those raw materials into the final product. Process planning is therefore the task of precisely specifying how to manufacture a particular product. As such, process planning forms the link between design and manufacturing.

PSL requirements for process planning

1.abstraction.
2.alternative task
3.associated illustrations and drawings - for explanatory purposes.
4.complex groups of tasks.
5.complex resource characteristics
6.complex sequencing of tasks.
7.complex task representation and characteristics
8.concurrent tasks
9.conditional tasks

10. confidence levels
11. constraints
12. date(s) and time(s) and/or multiple duration(s)
13. iterative loops.
14. manufacturing product quantity.
15. material constraints
16. parallel tasks
17. parameters and variables
18. pre- and post-conditions
19. resource categorization and grouping
20. resource/task combined attributes.
21. serial tasks
22. state existence constraints
23. state representations
24. temporal constraints
25. uncertainty/variability/tolerances

3. PETRI NET REPRESENTATIONS FOR MANUFACTURE PROCESS PLANNING

In this section we will demonstrate that some of the above requirements of the PSL may be successfully represented in a special class of Petri nets, the Compact Process Planning nets.

Petri nets

A Petri net consists of places and transitions which are linked to each other by directed arcs, with some arcs directed from places to transitions (input arcs), and some arcs directed from transitions to places (output arcs). A Petri net can be described as a bipartite directed graph whose nodes are a set of places and a set of transitions.

Places represent *passive* system components which store "tokens" and take particular states. Graphically, places are represented by circles.

Transitions represent the *active* system components which may produce, transport or consume "tokens". For each transition there is a set of input places and a set of output places. Graphically, transitions are represented by rectangles.

Arcs connect places with transitions and represent the relations between them. The arc's direction indicates the flow of information (token flow) through the net: from input places to transitions or from transitions to output places. Arcs have a weight which indicates the number of tokens that are transferred from an input place to the connected transition or from a transition to a connected output place.

A transition is enabled if in each of its input places there is a number of tokens greater or equal to the weight of the corresponding arc. When enabled, a transition removes a number of tokens, equal to the weight of the corresponding arc, from each input place and adds a number of tokens, equal to the weight of the corresponding arc, to each output place.

Marking or state of a Petri net is the position of tokens in the net at any instant in time. A given marking of a Petri net defines which transitions are able to fire at that time. The firing of a transition moves the net to a new marking.

A marking μ' is said to be reachable from a marking μ if there is a sequence of intermediate markings (and transitions) leading from μ to μ'. The set of all reachable markings from μ is called reachability set and can be represented by a reachability graph.

Representation 1: Compact Process Planning net

To construct the CPP-net of a process planning problem we first determine the necessary machining tasks and associated resources. All possible precedence relationships among machining tasks are recognized and established. This is very important because it confines the number of possible solutions. For the same reason, groups of tasks to be executed under the same conditions are established if possible.

The following rules are applied for the construction of the proposed Petri net model for process planning:

1. Each machining task, or a well established group of them, is represented by a transition $\mathbf{T_i}$, (i=1,...,n).
2. For each transition $\mathbf{T_i}$ there is one input place initially marked with one token. After firing a transition $\mathbf{T_i}$, the corresponding input place $\mathbf{IP_i}$, looses its only token. This indicates that the transition $\mathbf{T_i}$ cannot be fired again and, consequently, the represented task cannot be considered again.
3. Alternative (candidate) machining tasks, if any, are represented by transitions $\mathbf{T_{ij}}$ with common input and output places.
4. Precedence constraints are modeled with a common place, called constraint place $\mathbf{CP_{im}}$, between two transitions with a precedence relation. A constraint place $\mathbf{CP_{im}}$ is an output place for the ancestor transition $\mathbf{T_i}$ and an input place for the successor transition $\mathbf{T_m}$.
5. There is a common input-output (dynamic) place called **ControlPlace**, marked initially with one token, which is connected with bidirected arcs with all transitions. This place assures that, among the set of all enabled transition at a given time, only one can be fired. This construction satisfies the requirement that, within a usual manufacturing environment, only one task can be processed at a time.
6. All arcs are weighted by 1. This assures that only one token (task) will be considered for processing at each iteration.

A Petri net model constructed according to the above rules:

- represents accurately and dynamically the process planning procedure for a given mechanical part
- provides a graphic tool for knowledge representation of the type of precedence relations constraints, represented by the relation: transition-output place-successor transition
- provides a powerful simulation tool for process planning, simulating both machining task sequence and state of part and machine tool
- gives all possible solutions/process plans by simulation tracing or **reachability analysis**

Formal Definition of CPP-net

A Petri net N =[P, T, F, M_0] is a CPP-net (Compact Process Planning net) if and only if:

1. P is the set of places of the net. Places represent manufacturing precedence constraints or alternatives. There are three types of places: one place called *ControlPlace*; one set of input places $\boldsymbol{IP_i}$; and one set of constraint places $\boldsymbol{CP_{im}}$.
2. T is the set of transitions of the net. Transitions represent manufacturing tasks.
3. F is the set of arcs connecting places and transitions.
4. M_0 is the initial marking and has one token in the ControlPlace and one token in each one of the input places $\boldsymbol{IP_i}$.
5. M_F is the final marking and has only one token in the ControlPlace and it is the only dead –end of the net.

6. There is no dead transition at M_0.
7. N is a safe Petri net.

A run σ of a CPP-net is a firing sequence of transitions that transforms the initial marking M_0 into the final marking M_F.

The runs of a CPP-net represent, in a bijective way, all process plans, i.e. all feasible sequencing of the tasks of the part to be produced. The *reachability graph* of a CPP-net models all runs, i.e. all process plans of a CPP-net. The nodes of the reachability graph represent all possible states of the workpiece during the manufacturing process and the edges represent the transitions of the CPP-net, i.e. the machining tasks.

Figure 1. shows graphically the constructing elements and the basic generic structure of the proposed Petri net model for process planning (CPP-net).

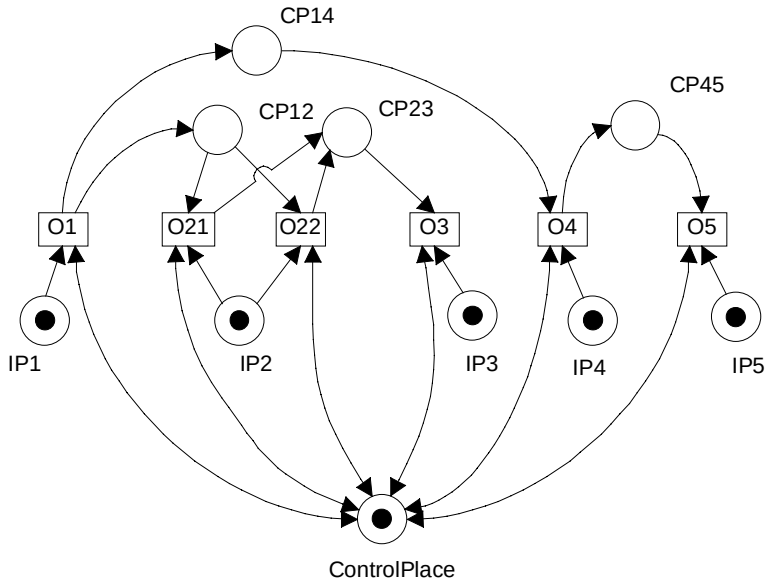

Figure 1. The CPP-net

Data structure of the CPP-net transitions

In order to model the cost of a process plan we have to attach the relevant information to our model. For process planning purposes, we may distinguish four types of cost: (i) the pure machining cost, (ii) the cost of moving a part from one machine to another, (iii) the cost of a setup change in one machine and (iv) the cost of a tool change in one machine. The pure machining cost depends mainly on the time a machine is used for a particular machining task. These costs are incorporated into the CPP-net by assigning to every task transition of the CPP-net the following information:

taskTransition := (task, (machine, (setup, tool))) (1)

Given the above, a state of the CPP-net is characterized by the following formula:

currentState := (tasksState, (currentMachine, (setup[i], tool[i]))) (2)

The *tasksState* is given by the marking of the CPP-net. It indicates which tasks are already done and what are the next possible tasks. The *currentMachine* is the one on which the last task was performed, while the pairs *(setup [i], tool[i])* for each machine *i*, characterize what is the current state of the corresponding setup and tool. The *enabling* of a task is determined by

the *tasksState* but the *cost* of that task depends also on the rest of the *currentState*, more precisely it is given by the cost of the task, plus a cost for a machine change, plus a cost for a setup change, plus a cost for a tool change, if these changes take place. If there is no machine, setup or tool change, the corresponding costs are null.

Grouping of tasks with CPP-nets

The *main task* is to manufacture the part. It is a *complex task* which is decomposed into a set of smaller *subtasks* considering various methods for accomplishing the task. By applying this concept, we say that *primitive tasks* cannot be broken down into any smaller tasks. In tese terms a machining operation in our case is a primitive task. Each complex task is also a group of its subtasks and a collection of tasks linked together is called a *hierarchical task network* or task network.

A group of tasks can be represented as a sub CPP-net named Group-net. The Group-net has two special transitions: G-begin and G-end. There is an arc from the control place to the G-begin transition and from the G-end transition to the control place. Each task transition of the group (e.g. T2, T3 and T4) has two places, one input and one output place. Each input place of the task transitions is also output place of the G-begin transition, while each output place of an task transition is an input place of the G-end transition. A Group-net can be simplified when there is a precedence constraint between two transitions of a group.(e.g. Fig 2 : between T3 and T4). In that case, we need to have one output place only for the last constrained transition.

We distinguish two kinds of groups: (i) closed groups which must finish before continuing with other tasks and (ii) open groups where tasks outside the group are allowed to fire together (in “competition”) with the tasks of the group.

Closed groups are created when there is a precedence constraint between the group and other transitions or groups (e.g. in Figure 2a between G-end and T5).

Open groups are created when the planner allows alternative technologies to be considered for a manufacturing feature. In that case, the control place is connected to the G-begin transition with a double arc like with any other task transition (e.g. in Figure 2b).

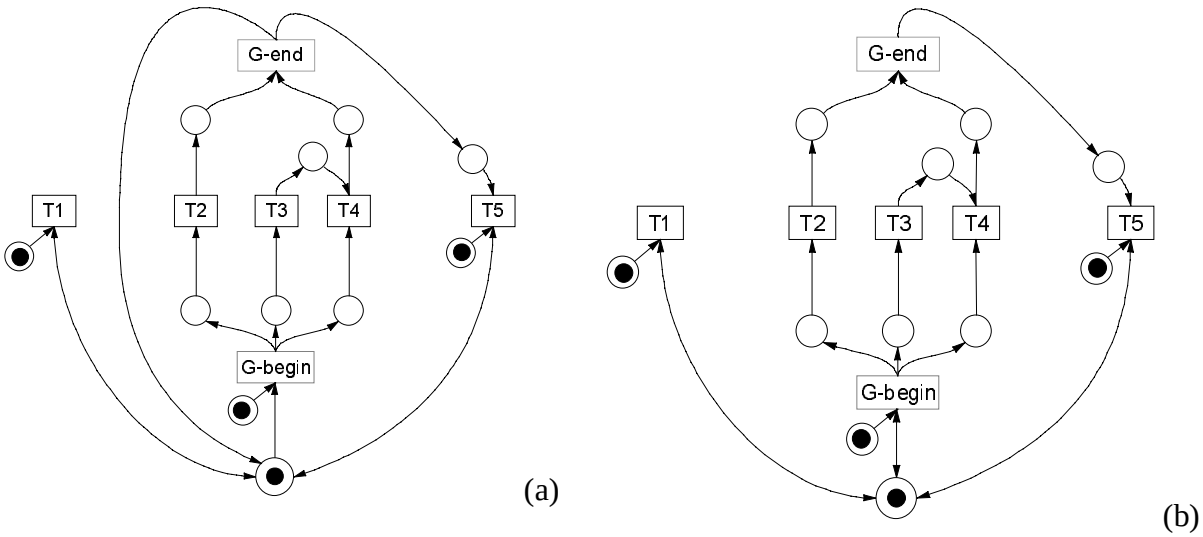

Figure 2. Groups of tasks with CPP-nets: (a) closed group, (b) open group

4. PSL MANUFACTURING SCENARIO 1: EDAPS MICROWAVE T/R MODULE

A microwave transmit / receive (T/R) module is an electrical component that can be found in modern telecommunication devices designed for scientific and commercial long-range defence applications (e.g. radar, satellite communications, long distance television and telephone signal transmissions). These modules are complex devices having both electrical and mechanical properties. For more information on the EDAPS scenario, see [Smith, 1997].

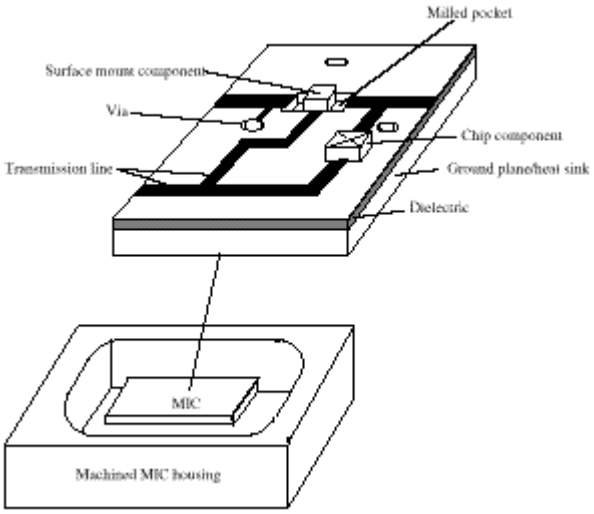

Figure 6. A typical microwave T/R module : the MIC substrate, and its housing

Electronic Processes

One method for making the artwork for the MIC (figure 6) is to do the following series of tasks : precleaning for the artwork, then application of photoresist, then photolithography for the artwork, then etching. There are several methods for applying photoresist : spindling photoresist, spraying on the photoresist, painting on the photoresist, and spreading out the photoresist from a spinner.(figure 7).

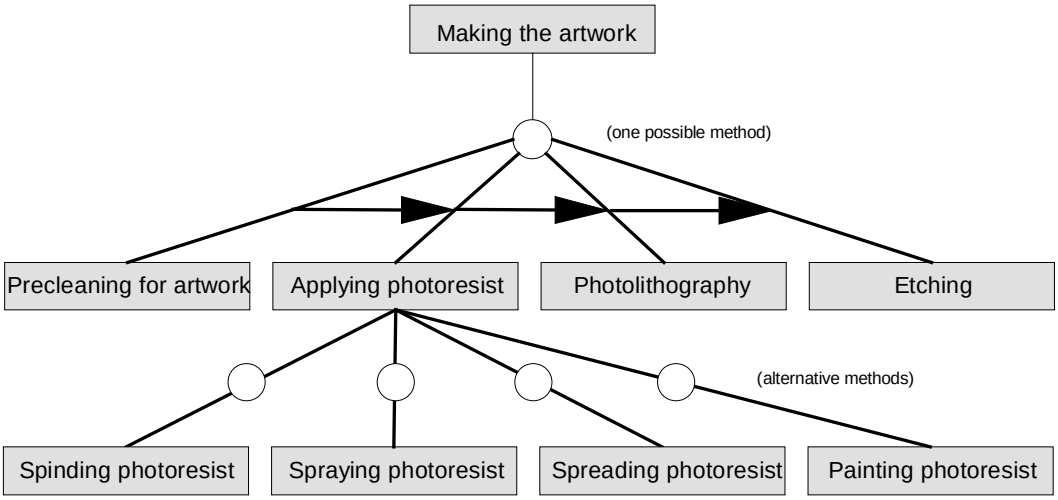

Figure 7. Electronic part of the task network for microwave T/R module manufacture

Mechanical Features

To manufacture a microwave T/R module A, a process plan needs to be generated for the following design and environment you can see on figure 8. A board, E, of 4.00 inches long, 6.00 inches wide, and 1.00 inch thick will be used. Board E will be made up of two layers, the ground plane and the dielectric. The ground plane is made up of copper. The dielectric material is Teflon and has resistivity 0.10. A plated through-hole needs to be made at (0.25,5.88). The through-hole will be of diameter 0.20 inches, will have a depth of 1.00 inch, and will have plating of width 0.10 inch. A hole of diameter 0.40 inches and depth of 1.00 inch is needed at (0.50,0.25). There will be two milling features, C and D, at (0.00,2.25) and (1.20,4.20) respectively. Both of them will not be rotated. Milling feature C will be 0.50 inches deep, 0.40 inches long, and 0.30 inches wide. C will have a corner radius of 0.15. Milling feature D will be 0.50 inches deep, 0.30 inches long, and 0.40 inches wide. It will have corner radius 0.08. There is a minimum gap of 10.00 mils in the artwork. There will be one component, C1, on the artwork. The component is to be placed at (3.60,5.00). C1 is 0.35 inches long, 0.30 inches wide, and 0.20 inches high. It can be simply inserted without any adhesive. C1 has two pins, each of which at (0.17,0.00) and (0.17,0.30).

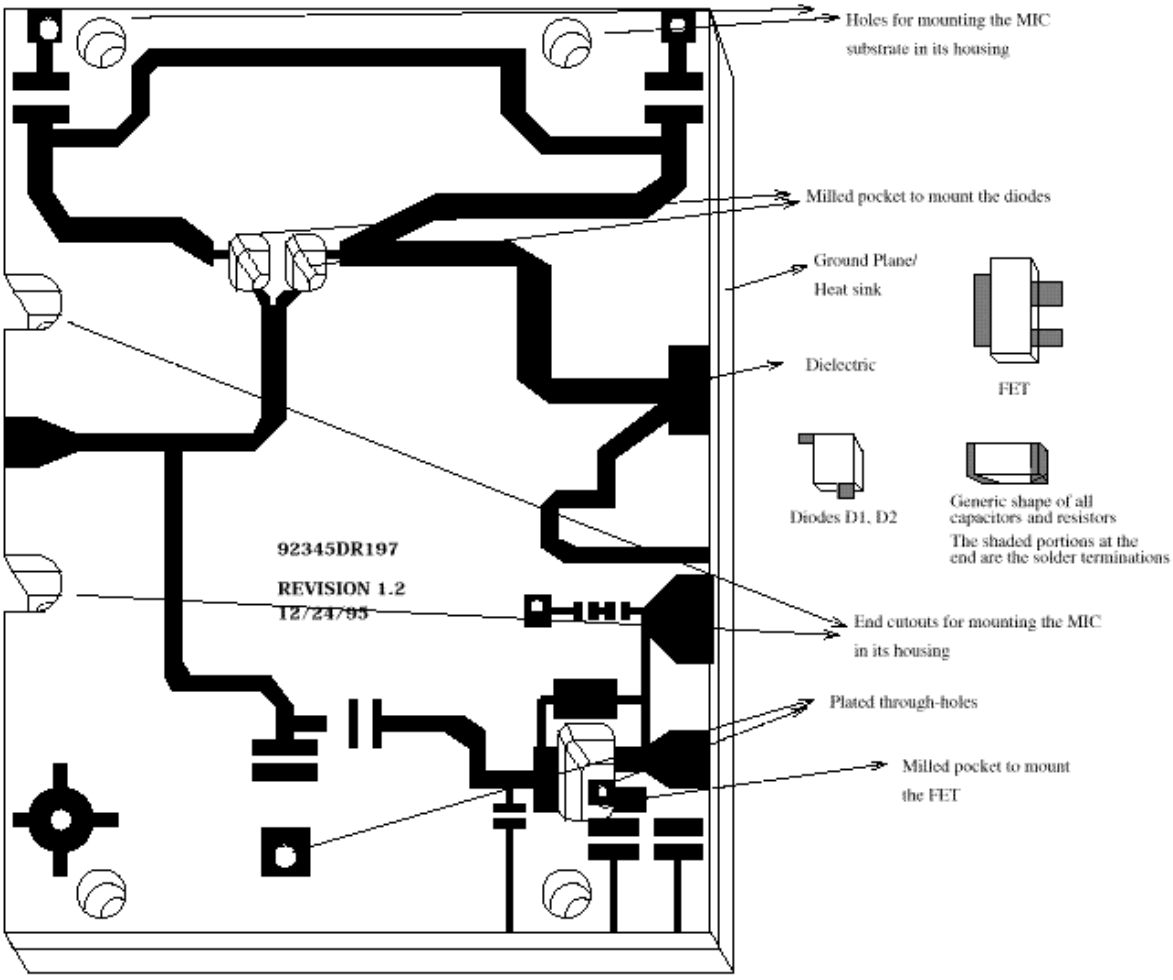

Figure 8. Development of mechanical features on the Mixer-IF amplifier substrate

Task description

The process plan is expected to perform under a time constraint of 20 minutes. A maximum of 10 workers will be provided to accomplish the manufacturing. Finally, one of

each of the following machines will be available throughout the manufacturing. They are VMC1 (Vertical Machining Centre), EC1 (Electrical Centre), PC1 & PC2 (Platting Centres), MC1 (Manufacturing Centre), TC1 (Testing Centre), HSC1 (Hermetical Sealing Centre), and AC1 (Adhesive Centre). A process plan that requires the least amount of time will be most preferable.
The process planner is restricted to the following performable tasks:

tasks	chosen machine	Steps/alternatives	Setup time	mach. time
i. Drill plated through-holes	VMC1	setup : orient table, clamp board, establish datum point : *A1.1.1.1 (O1)*	2	0
		Install drill Ø0.3 and rough drill : *A1.1.2.1 (O2)*	0.1	0.43
		Finish drill : *A1.1.3.1 (O3)*		
		Install drill Ø0.2 and rough drill 4 times :*A1.1.3.1 (O4..O7)*	0.1	0.77
		Finish drill 4 times : *A1.1.4.1 (O8..O11)*		
ii. Plate through-holes	PC1	condition board : *A2.1.1.1 (O12)*	0	10
		Pickle board : *A2.1.2.1 (O13)*	0	10
		Activate board : *A2.1.3.1 (O14)*	0	10
		Electroless copper deposition : *A2.1.4.1 (O15)*	0	15
iii. Platting resist	PC2	Apply platting resist : *A3.1.1.1 (O16)*	30	0.5
		Copper electroplating : *A3.1.2.1 (O17)*	0	25
		Tin/Lead electroplating : *A3.1.3.1 (O18)*	0	15
		Strip plating resist : *A3.1.4.1 (O19)*	0	5
iv. Pre-clean the board	EC1	*A4.1.1.1 (20)*	0	32.29
v. Application of phtoresist	EC1	spindle photoresist : *A5.1.1.1 (O21)*	30	0.4
		Spray photoresist : *A5.1.1.2 (O22)* *alternatives (Alt1)*	30	0.45
		Spread photoresist : *A5.1.1.3 (O23)*	30	0.45
		Paint photoresist : *A5.1.1.4 (O24)*	30	0.65
vi. Photolithography of photoresist	EC1	*A6.1.1.1 (O25)*	30	2.2
vii. Etching	EC1	*A7.1.1.1 (O26)*	30	28
viii. Make slots, pockets, and holes	VMC1	setup (orient board, clamp board, establish datum point) : *A8.1.1.1 (O27)*	2	0
		Install side-milling toll, Rough side-mill pocket 1 : *A8.1.2.1 (O28)*	0.1	0.34
		Finish side-mill pocket 1 : *A8.1.3.1 (O29)*		
		Rough side-mill pocket 2 : *A8.1.4.1 (O30)*		
		Finish side-mill pocket 2 : *A8.1.5.1 (O31)*		
		Install end-milling tool, rough end-mill pocket 1 : *A8.1.6.1 (O32)*	0.1	1.54
		Rough end-mill pocket 2 : *A8.1.7.1 (O33)*		
		Install slot-milling tool, rough slot mill pocket 1 : *A8.1.8.1 (O34)*	0.1	1.24
		Install drill Ø0.4, rough drill hole : *A8.1.9.1 (O35)*	0.1	1.54
		Rough drill hole : *A8.1.10.1 (O36)*		
		Rough drill hole : *A8.1.11.1 (O37)*		
		Rough drill hole : *A8.1.12.1 (O38)*		
		Install drill Ø0.15, rough drill hole : *A8.1.13.1 (O39)*	0.1	0.22
		Finish drill hole : *A8.1.14.1 (O40)*		
		Rough drill hole : *A8.1.15.1 (O41)*		
		Finish drill hole : *A8.1.16.1 (O42)*		
ix. Pre-clean before soldering	MC1	*A9.1.1.1 (O43)*	30	5.71
x. Screenprint solder stop to board	MC1	*A10.1.1.1 (O44)*	30	0.29
xi. Apply solder paste to board	MC1	30 times (group) : *A11.1.1-30.1 (O45-O74)*	30	7.5
xii. Dry solder paste	MC1	*A12.1.1.1 (O75)*	0	5.71
xiii. Manual pin transfer of adhesive to board	AC1	L1, L2 & FET(group) : *A13.1.1-3.1 (O76-O78)*	30	0
xiv. Heat curing of adhesive	AC1	*A14.1.1.1 (O79)*	30	10
xv. Pick and place	MC1	15 times (group) : *A15.1.1-15.1 (O80-O95)*	0	7.5
xvi. eflow soldering	MC1	*A16.1.1.1 (O96)*	30	5
xvii. Hand soldering	MC1	L1, L2 & FET (group) : *A17.1.1-3.1 (O97-O99)*	0	7
xviii. Flux cleaning	MC1	*A18.1.1.1 (O100)*	0	11.43
xix. Pre-cap testing	TC1	*A19.1.1.1 (O101)*	0	35
xx. Shielding/hermetically sealing	HSC1	*A20.1.1.1 (O102)*	0	1.86
xxi. Post-cap testing	TC1	*A21.1.1.1 (O103)*	0	35
xxii. Final inspection	TC1	*A22.1.1.1 (O104)*	0	29.67

Figure 9 : Description of the EDAPS tasks

For each *task*, the planner choose the *technology* alternatives, before describing the necessary *steps*. Then, he chooses the available machines and defines the setups and the tools. He can allow several *alternatives*. As described before:

- Tasks represent things you want to accomplish. Therefore, a *task* is usually a geometry to make ; it can also be a group of geometric entities to make, an auxiliary task (i.e. cleaning), etc.
- A technological alternative is an alternative of technological possibilities for a task; a group is automatically created for each *technology*. On this example, technological alternatives are not defined
- For a chosen technology, there is a list of *steps* needed to realise the task at the defined level of quality
- An *alternative* is a choice among possible alternative solutions for a primitive task.

The EDAPS scenario proposes a list of tasks (figure. 9) to manufacture a microwave T/R module A. At these stage, no alternative of processes are defined between the chosen tasks. A list of precedence constraints is set from task (i) to task (xxiii). However, there's an alternative for the application of photoresist (v) with four alternatives (O21, O22, O23 &O24). Some groups are already created when the manufacturing times are calculated for a group of tasks in (i), (viii), (ix) (xii), (xiv), (xvi) & (xviii), and the process is defined as groups of tasks on the same machine. Because of that, the optimal processes are already given, in some way.

Grouping the described steps :

Groups are created when there's no precedence constraints between several tasks, or when we have a precedence constraint between a component and several other ones (figure 10).

	Name	***Components***	***Description***
(i)	G1	O15-O18	geometrical tasks (side-milling vs same tool/setup/machine)
(ii)	G2	O19-O20	geometrical tasks (end-milling vs same tool/setup/machine)
(iii)	G3	O22-O25	geometrical tasks (drilling vs same tool/setup/machine)
(iv)	G4	O26-O29	geometrical tasks (drilling vs same tool/setup/machine)
(v)	G5	O32-O61	same tasks vs different positions
(vi)	G6	O63-O65	same tasks vs different positions
(vii)	G7	O67-O82	same tasks vs different positions
(viii)	G8	O84-O86	same tasks vs different positions
(ix)	G9	O4-O7	same tasks vs different positions
(x)	G10	O8-O11	same tasks vs different positions
(xi)	G11	G9, G10	same machine
(xii)	G12	O12-O15	generic task
(xiii)	G13	O16-O19	generic task
(xiv)	G14	G1, G2, G3, G4, O34	same machine

Fig. 10 : List of the chosen groups of the EDAPS scenario

In the EDAPS scenario, primitive tasks are grouped either by manufacturing tools (i), (ii), (iii) & (iv), when several tasks can be grouped in a complex task, (xii) & (xiii) or when there are similar tasks to achieve at different positions (v), (vi), (vii), (viii), (ix) & (x). Two groups are created when several groups depend on the same machine (xi) & (xiv).

Constraints :

We distinguish two kinds of constraints : the first ones are evident because the planner choose steps to make a task, the second ones are explicitly given by the planner.
The elementary constraints are set between two alternatives or groups. We can drawn a list of the whole precedence constraints (figure 11):

	Precedence constraint	***Description***
(i)	O1→(O2; G9; G10)	setup before machining (VMC1)
(ii)	O2→O3	rough drill before finish drill (VMC1)
(iii)	G9→G10	rough drill before finish drill (VMC1)
(iv)	G11→G12→G13→O20	task constraint
(v)	O20→Alt1 (O21..O24)	constraint between a task and an alternative of tasks
(vi)	Alt1 (O21..O24) → O25	constraint between an alternative of tasks and a task
(vii)	O25→O26→O27→G14→O43	task constraint
(viii)	O43→O44→G5→O75→G6	task constraint
(ix)	G6→O79→G7→O96→G8	task constraint
(x)	G8→O100→O101→O102	task constraint
(xi)	O102→O103→O104	task constraint
(xii)	O28→O29; O30→O31	rough side-mill pocket before finish side-mill pocket
(xiii)	O39→O40; O41→O42	rough drill before finish drill
(xiv)	G1→G2 :	side-mill pocket before end-mill pocket
(xv)	G2→O34	end-mill pocket before slot-mill tool
(xvi)	O34→ G3	slot-mill tool before drill holes
(xvii)	G3→G4	drill hole diam 0.4 before drill hole diam 0.15

Fig. 11 : The list of the precedence constraints between the tasks of the EDAPS scenario

In the EDAPS scenario the tasks are subdivided in steps : setup and installing tools are made before machining (i), roughing is always made before finishing (ii), (iii), (xii) & (xiii). There is a list of constraints between every task (iv), (v), (vi), (vii), (viii), (ix), (x) & (xi). These constraints can be discussed and may be modified. We also can set different constraints between the tasks made on VMC1 (xiv), (xv), (xvi) & (xvii). Finally, there is an alternative of task for the application of photoresist (v) & (vi).

5. CPP-NET REPRESENTATION OF THE EDAPS SCENARIO

The CPP-net representation of the EDAPS scenario provide the basis for representing this simple processes. The information of time, resource and activity is included behind all the task/transition (e.g. O1..O104). Moreover, the CPP-net representation allows the description of :

(i) temporal/precedence constraints between the tasks. It is possible with the notion of input/output place and the precedence of tokens needed to fire the tasks/transitions
(ii) resource grouping (e.g. G1..G14)
(iii) alternative tasks (e.g. O21..O24)
(iv) alternative technologies (not used in the EDAPS scenario)

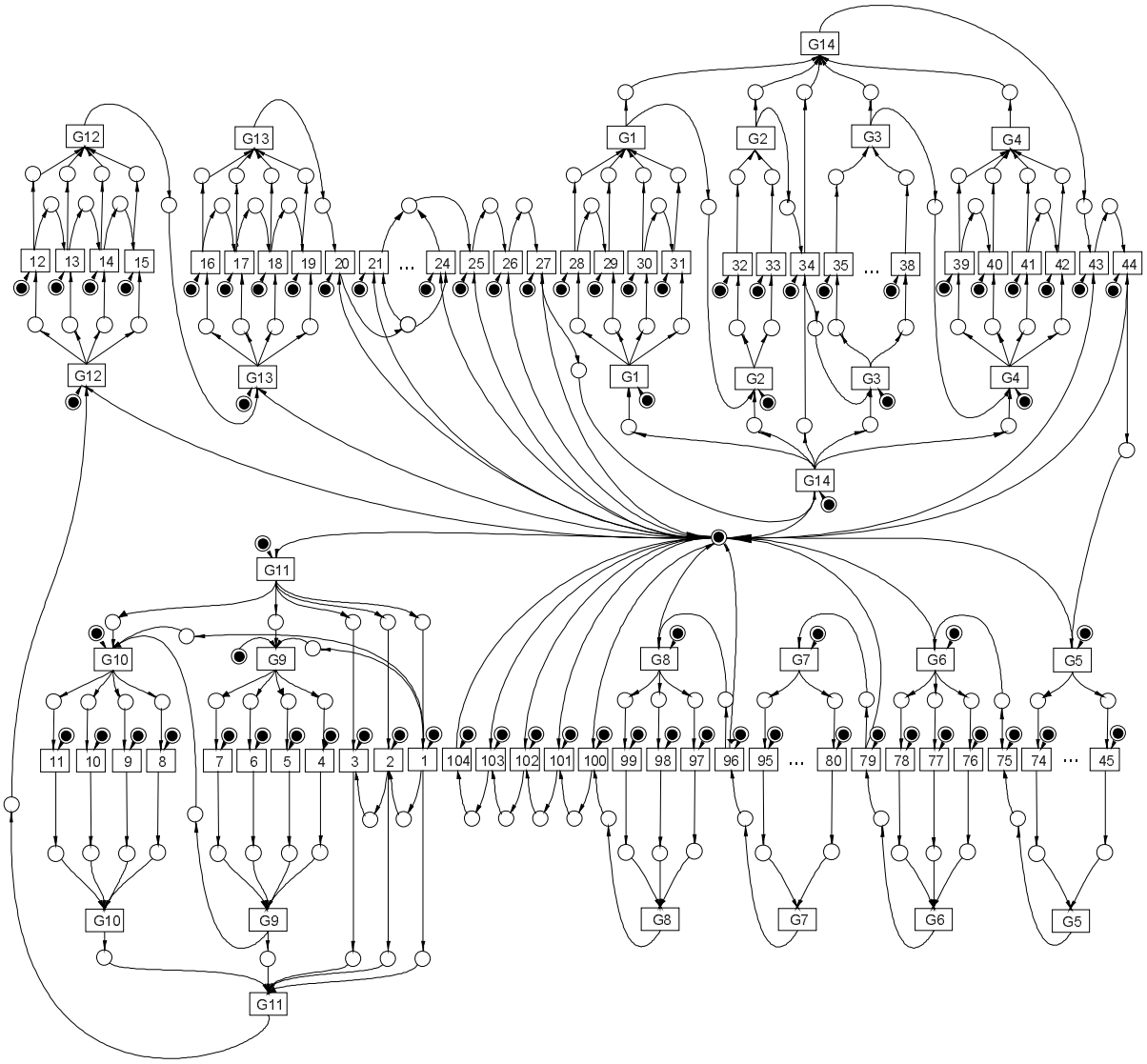

Fig. 12 : CPP-net representation of the EDAPS scenario : exploded view

The token in the control place represent the part. The token can move and fire all the task/transition which don't have any input place. In the EDAPS scenario the only one is the transition G11.

In the higher level grouped tasks can be represented by a simple transition. (e.g. G12)The graph at figure 12 is an exploded graph at the lower levels. The hierarchical network of the CPP representation is described in figures 13 and 14, the last one is the higher level.

Parallel tasks are grouped into the same task/transition; the graph you can see represent the serial tasks. We decided that only one transition/task can be fire at the same time.

Each task/transition can fire only once : they all have an input place with one token.

The notion of alternative processes is given when the token of the control place (the part) can fire different task/transitions. In the EDAPS scenario, as the global process (complex tasks are ordered) is given, this case doesn't appear very often.

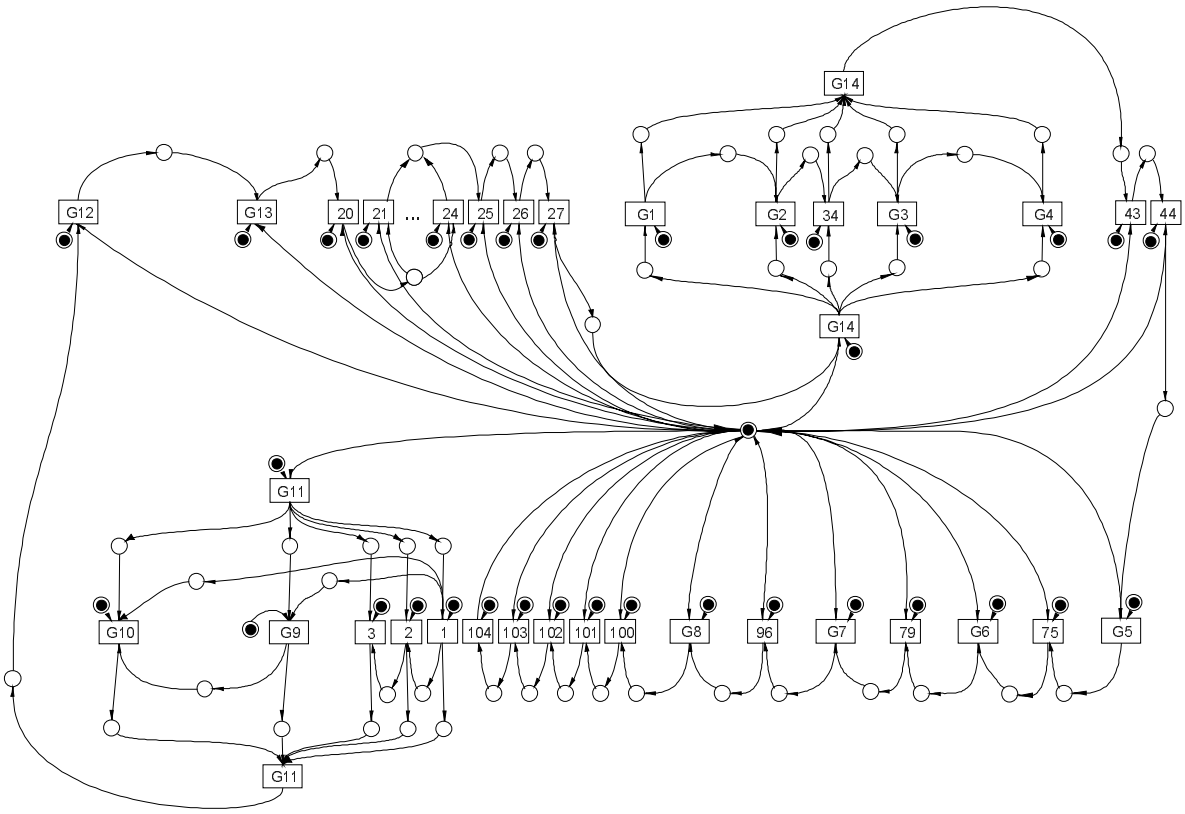

Fig. 13 : CPP-net representation of the EDAPS scenario : 2nd level view

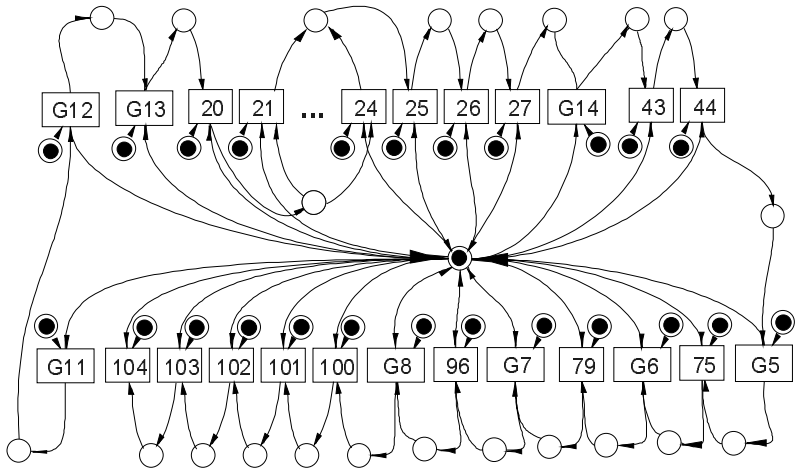

Fig. 14 : CPP-net representation of the EDAPS scenario : high level view

6. CONCLUSIONS

In this paper we propose Petri nets as a representation technique for modelling and analysing process planning activities as they are specified in the Process Specification Language (PSL) project of NIST. The goal of this project is to identify or create a process specification language (PSL) that can be common to all manufacturing applications, generic enough to be decoupled from any given application, and robust enough to be able to represent the necessary process information for any given application. The model discussed in this paper

allows the modelling of tasks, their precedence constraints and the required resources (machines, setup and tools) and the corresponding costs within a compact Petri net, named Compact Process Planning net (CPP-net). On the proposed example, our methodology of process representation can seem heavy. In reality, it allows essentially to envisage technologies and production alternatives. So, it lets more flexibility later to the scheduling service to plan the workshop. It allows the technological and specific group establishing defined by the Bureau of Methods. Precedence Constraints are introduced, either automatically referring some to successive steps, or manually by the Manufacturing Manager. Finally the obtained graph represents the totality of solutions of process planning a product.

7. REFERENCES

CECIL, J.A., SRIHARI, K., EMERSON, C.R., 1992, A review of Petri Net Applications in Process Planning, The International Journal of Advanced Manufacturing Technology, 7:168-177.

CHRYSSOLOURIS, G. and CHAN, S., 1985, An integrated approach to process planning and scheduling, Annals of the CIRP, Vol. 34(1), pp. 413-415.

HAM, I., LU, S. C.-U., 1988, Computer-Aided Process Planning: The Present and the Future, CIRP Annals, 37/2:591-601.

KIRITSIS, D. and PORCHET, M., 1996, A generic Petri net model for dynamic process planning and sequence optimisation, *Advances in Engineering Software*, Vol. 25, No. 1, pp. 61-71.

KIRITSIS, D., XIROUCHAKIS, P., 1996, A Software Prototype for Cost Estimation of Process Plans of Machined Parts**,** ISATA'96/Mechatronics, Florence, 19:26.

KRUTH, J.P., DETAND, J., 1992, A CAPP System for Nonlinear Process Plans, Annals of the CIRP, 41/1:489-492.

SCHLENOFF, C., KNUTILLA, A., RAY, S., 1997, Requirements for Modeling Manufacturing Process: A New Perspective, Proceedings of the DETC'97: 1997 ASME Design Engineering Technical Conferences, September 14-17, 1997, Sacramento, California.

SMITH, S.J.J., 1997, Task-Network Planning using Total-Order Forward Search and Applications to Bridge and to Microwave Module Manufacture, Ph.D. Thesis, UMD.

SRIHARI, K., EMERSON, C.R., 1990, Petri Nets in Dynamic Process Planning, Computers Industrial Engineering, 9:447-451.

TÖNSHOFF, H. K., BECKENDORFF, U., ANDERS, N. and DETAND, J., 1991, A Process Description Concept for Process Planning, Scheduling and Job Shop Control, *Proceedings of the CIRP Seminar, Manufacturing Systems*, Vol. 20, No. 1, pp. 53-60.

TÖNSHOFF, H. K., KREUTZFELD, J. and HOFSCHNEIDER, D., 1993, Concurrent process planning and workshop control in batch production-load oriented process planning, *Proceedings of the CIRP Seminars, Manufacturing Systems*, Vol. 22, No. 3, pp. 231-241.